doi:10.11887/j.cn.202502013 http://journal.nudt.edu.cn

多芯粒网络中负载均衡的死锁解决算法

周宏伟[1,2],陈志强[1,2*],曾 坤[1,2],邓让钰[1,2]
(1. 国防科技大学 计算机学院,湖南 长沙 410073;
2. 国防科技大学 先进微处理器芯片与系统重点实验室,湖南 长沙 410073)

摘 要:针对多芯粒网络中存在跨芯粒的死锁问题以及链路故障导致的网络连通性问题,提出一种面向多芯粒网络的优化报文重传机制。通过在重传机制中使用“报文合并”功能来减少控制报文的数目,降低网络的负载;通过使用“报文转发”功能并采用转发到邻居策略,降低芯粒间网络链路故障的容错成本,实现芯粒内网络更均衡的负载。模拟实验结果表明:相较于转向限制,所提方法在延迟基本不变的前提下提升12.5%~25%的饱和带宽,在出现链路故障时,最高提升50%的饱和带宽。“报文合并”可以有效减少控制报文的数目从而降低网络整体的负载。“报文转发”容错成本更低、能够实现芯粒内网络更均衡的负载。

关键词:芯粒;容错;死锁;重传
中图分类号:TP303 **文献标志码**:A **文章编号**:1001-2486(2025)02-146-09

论文拓展

Load-balanced deadlock resolution algorithm in multi-chiplet network

ZHOU Hongwei[1,2], *CHEN Zhiqiang*[1,2*], *ZENG Kun*[1,2], *DENG Rangyu*[1,2]
(1. College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China;
2. Key Laboratory of Advanced Microprocessor Chips and Systems, National University of Defense Technology, Changsha 410073, China)

Abstract: To solve the inter-chiplet deadlock and network connectivity problems caused by link failures in multi-chiplet network, an optimized packet retransmission mechanism for multi-chiplet network was proposed. By using the "message merging" function in the retransmission mechanism, the number of control packets and the network load was reduced. By using the "message forwarding" function and adopting the forwarding to neighbor strategy, the fault-tolerant cost of the inter-chiplet network link failure was reduced. And more balanced load of the intra-chiplet network was realized. The simulation results show that the proposed method can increase the saturation bandwidth by 12.5%~25% with similar latency compared with the turn restriction strategy. Furthermore, it can increase the saturation bandwidth by up to 50% in case of link failures. "Message merging" can effectively reduce the number of control packets, thus reduce the overall load of the network. "Packet forwarding" has lower fault-tolerance cost and can achieve more balanced load of intra-chiplet network.

Keywords: chiplet; fault-tolerance; deadlock; retransmission

由于单芯片晶体管数量受到工艺发展极限、良率及成本等因素的限制,片上系统(system on chip,SoC)设计所采用的在单芯片上集成更多晶体管的技术路线遇到瓶颈。将SoC划分为若干芯粒(chiplet),使用2.5D封装技术将这些芯粒连接起来,构成基于芯粒的系统[1-5],成为进一步提升芯片规模和性能的主要方式之一。相较于传统的SoC,芯粒的生产成本和设计难度更低,并且具有良好的复用性。可以根据设计需求,封装不同功能、不同生产工艺的芯粒,快速搭建芯片[6-9]。近年来,在学术界,基于芯粒构建系统的关键技术研究成为热点,工业界[10-13]部分基于芯粒的产品也已经投入商用。

将多个无死锁的芯粒连接起来,构成的多级网络存在两个新的问题。

首先是会产生新的死锁[14-17]问题。芯粒内

收稿日期:2022-12-12
基金项目:国家部委基金资助项目(31513010502)
第一作者:周宏伟(1980—),男,陕西宝鸡人,研究员,博士,硕士生导师,E-mail:zhou.hongwei@139.com
***通信作者**:陈志强(1998—),男,河南开封人,博士研究生,E-mail:czq20@nudt.edu.cn
引用格式:周宏伟,陈志强,曾坤,等. 多芯粒网络中负载均衡的死锁解决算法[J]. 国防科技大学学报,2025,47(2):146-154.
Citation: ZHOU H W, CHEN Z Q, ZENG K, et al. Load-balanced deadlock resolution algorithm in multi-chiplet network[J]. Journal of National University of Defense Technology, 2025, 47(2): 146-154.

以及芯粒间的网络均采用 mesh 网络拓扑结构,采用 *XY* 路由算法[14,18]。*XY* 路由算法通过限制严格的维度顺序来避免死锁,在商业处理器中广泛使用。尽管每个芯粒内部的片上网络(network-on-chip, NoC)以及多个芯粒之间的互联网络各自都不产生死锁,但是经过层次化连接后,却可能会产生死锁。由于这种死锁涉及多个芯粒 NoC 网络,传统的死锁解决算法无法很好发挥作用。

Yin 等提出了模块化转向限制(modular turn restriction,MTR)方法[14],通过在边界节点上对跨芯粒的报文施加转向限制来避免死锁。MTR 遍历可能的转向限制和边界节点位置的组合,在保证通道依赖图不会形成闭环的前提下,从可用选择中选取最优解。添加转向限制存在以下问题:①降低路径多样性,可能会增加延迟以及降低带宽[18];②可能导致边界节点之间的负载不均衡;③转向限制和边界节点位置组合的遍历复杂度较高,当某处芯粒间链路出现故障后,可能需要重新配置转向限制,容错成本高。

Majumder 等提出了远程控制(remote control, RC)方法[15],依据目的节点位置的不同,将报文划分为芯粒内(intra-chiplet)和芯粒间(inter-chiplet)两类。通过搭建的控制网络调控每个芯粒内跨芯粒报文的注入,使通道依赖关系无法通过边界节点传递到其他芯粒,避免通道依赖图中出现环。该方法的缺点是:①由于流控机制的存在,跨芯粒报文注入网络之前需要和边界节点完成一次交互,会增加报文的延迟;②当出现芯粒间链路故障后,需要重新构建控制网络,提高了系统的容错成本。

Wu 等在分析跨芯粒的死锁场景时,观察到死锁闭环中包含一类报文上升报文,提出将这类报文通过垂直的上升通道直达目的节点所在芯粒的上升报文弹出(upward packet popup, UPP)方法[16]。UPP 通过旁路机制搭建一条虚拟通道,使得上升报文绕过中间节点直达目的节点[19],从而从死锁中恢复。该方法不需要限制跨芯粒报文的注入或者传输路径,具有更好的容错性和灵活性,然而设计实现相对复杂。

通过分析,以上方法在网络延迟和网络带宽优化、芯粒内网络边界节点的负载均衡、芯粒间网络链路发生故障后的容错性等方面各自存在一些不足。

其次是跨芯粒的链路发生故障导致的带宽降低、容错和负载不均衡问题。在芯粒系统中,跨芯粒的报文传输都需要经过边界节点,边界节点的数目和位置会影响跨芯粒传输的最大带宽。在 $N \cdot N$的 mesh 网络中,N 个边界节点就可以满足跨芯粒报文传输的带宽要求[14]。边界节点的数目通常少于芯粒中的路由节点数目,意味着每个边界节点要负责传输多个路由节点的跨芯粒报文。边界节点出现故障,会影响芯粒中多个路由节点的跨芯粒报文传输、芯粒间报文拥塞、边界节点的负载增加以及不均衡等问题。静态绑定(static binding)策略静态确定边界节点,芯粒上的路由节点对应唯一的边界节点[14-16]。静态绑定策略易于实现,但是也存在容错性较差、容错成本高、负载不均衡的问题。

通用芯粒互联技术(universal chiplet interconnect express,UCIe)中的先进封装模式支持通道重新映射,可以提高容错性,但可能会造成通道的带宽降低[20]。相关工作发现,死锁和拥塞之间的正反馈关系[21],即无法及时恢复的死锁会加剧网络的拥塞,而拥塞的网络中更易于产生死锁。这些问题需要在路由算法层面解决,在解决死锁的同时,提高芯粒系统容错性、降低容错的成本、提升链路故障情况下的系统性能。

针对以上不足,提出一种面向芯粒网络的优化的报文重传机制以解决跨芯粒的网络死锁问题。报文重传机制实现难度低,不需要修改一致性协议,不受路由算法、拓扑结构的限制[22-23],具有良好的可移植性。考虑到重传机制会新增控制报文进而增加网络的负担,对网络延迟和带宽造成影响,直接使用收益较低,提出通过在重传机制中使用“报文合并”功能减少控制报文的数目,降低网络的负载。针对传统方法的边界节点分配算法在复杂度和灵活度方面无法权衡、容错性差或者实现成本高、负载不均衡等问题,在重传机制的基础上,通过使用“报文转发”功能并采用一种介于静态绑定和动态选择之间的边界节点分配策略,即转发到邻居(forward to neighbor, FTN)策略,提高芯粒间网络中链路故障的容错性,降低容错成本,实现芯粒内网络更均衡的负载。

1　面向芯粒网络的报文重传机制

在基于芯粒的系统中,多个芯粒通过中介层进行连接,可以将网络划分为芯粒网络和中介层网络。在基于芯粒的系统中,报文传输的流程通常如下:芯粒内报文在芯粒网络中传输;跨芯粒的报文需要首先到达边界节点(boundary router),然后经由中介层网络到达目的节点所在的芯粒,最终经由芯粒网络到达目的节点。

1.1 多芯粒网络死锁

基于芯粒的系统中的死锁涉及多个芯粒，不能直接应用传统的死锁解决算法。目前相关研究主要目标为解决多芯粒网络中的死锁问题，容错性作为死锁解决算法的重要特性，也受到广泛的关注[24-27]。图 1 为基准系统架构。图 1 中红色连线展示了一条死锁依赖环，整条路径部分位于芯粒内部、部分位于芯粒间的中介层网络。箭头表示通道之间的依赖关系，首尾相接形成了闭环，从而导致闭环中的所有报文都无法移动，形成死锁。

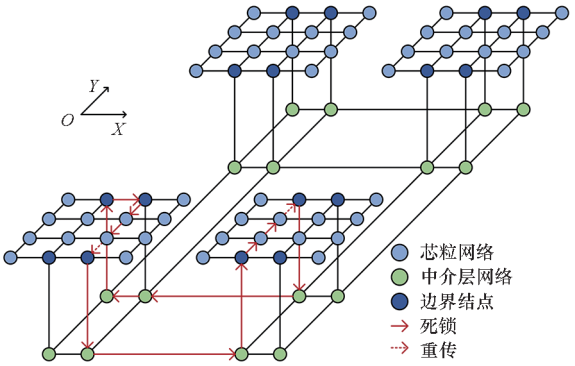


图 1 基准系统架构

Fig. 1 Baseline system architecture

图 2 给出了基于 MTR 的一种边界节点分配方案，用于说明在负载均衡和容错性方面的不足。图 2 为 MTR 下的报文分配情况，红色箭头表示额外的报文转向限制，绿色区域代表边界节点的控制范围，面积越大表示负载越重，可以看出部分边界节点承担了较多的负载，导致负载不均衡。

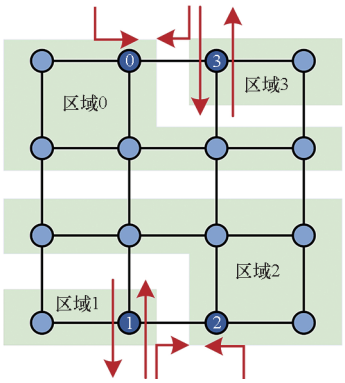


图 2 静态绑定的边界节点分配方案

Fig. 2 Boundary router allocation strategy for static binding

提出一种面向芯粒网络的报文重传机制，以解决跨芯粒的网络死锁问题。主要思想是：在由芯粒构成的多级网络中，死锁的产生是由于形成了跨芯粒的依赖关系，构成的死锁闭环中必定包含边界节点。如果跨芯粒的报文在抵达边界节点时没有被吸收，就返回重传报文，进行报文的重新传输；如果被吸收，就返回确认报文。由于报文不会堵塞，不会构成依赖关系，从而解决死锁。

1.2 报文重传机制工作原理

报文注入：报文在源节点注入芯粒网络前，会根据报文路由计算的结果，判断报文是否为跨芯粒传输。对于跨芯粒的报文，会在重注入缓冲中保留一个备份，用以实现报文的重传。当重注入缓冲为满时，跨芯粒的报文不能注入芯粒网络中，而芯粒内部的报文传输不受影响。

报文传输：跨芯粒报文会维护两个路由表，分别用于在芯粒网络和中介层网络中的传输。跨芯粒的报文注入芯粒网络后，会按照当前芯粒网络的路由算法到达指定的边界节点，这个过程中跨芯粒的报文和芯粒内的报文具有相同的传输模式。当到达边界节点后，会根据当前节点的状态选择返回重传报文或者确认报文。

确认报文：确认报文代表报文已经被边界节点吸收。如图 3 所示，P1 为跨芯粒的报文，从节点 4 出发，根据 *XY* 路由策略到达节点 1，假设 P1 被节点 1 吸收，则需要向源节点 4 返回确认报文。确认报文的生成由重传单元负责，重传单元内部存储着路由信息和 ID 号。路由信息用以确定确认报文的目的节点；ID 号可以唯一确定在源节点中保留的报文备份。

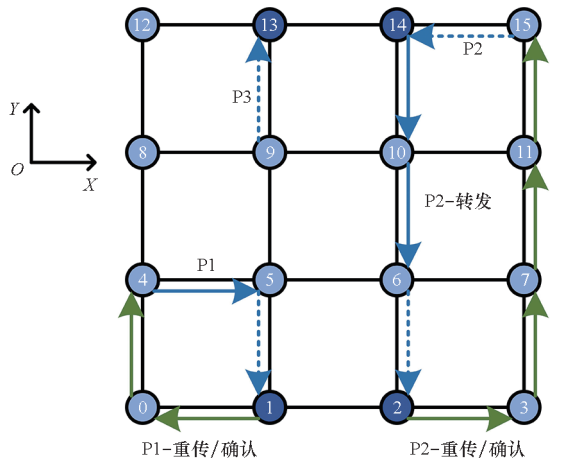


图 3 跨芯粒报文传输流程

Fig. 3 Transmission flow of inter-chiplet packets

重传报文：图 3 中，假设 P1 没有被节点 1 吸收，可能的原因如下。一是由于拥塞，暂时没有空闲的缓存资源，若干周期后，拥塞会逐渐改善，P1 最终会被吸收。二是由于死锁，形成了死锁闭环。

为了解决死锁,当 P1 阻塞时间超过阈值 T_{th} 时,会向源节点返回重传报文,进行报文的重新传输。重传报文的生成和确认报文的生成类似,不同在于源节点收到重传报文后,会根据 ID 号唯一确定报文的备份,将其重新注入芯粒网络中,重复之前的传输过程。如图 3 中 P1 所示,跨芯粒报文在规定阈值内没有离开节点 1,则丢弃堵塞的报文,向源节点返回重传报文。因此跨芯粒的报文总能在节点 1 被吸收,无法形成节点 5 和节点 1 直接的依赖关系,无法形成通道依赖闭环。报文的重传打破了节点 5 和节点 1 之间的通道依赖关系,从而解决死锁。

1.3　路由器节点架构

根据提出的重传机制,设计了一种路由节点架构。路由节点的实现如图 4 所示。相较于传统的路由节点,主要增加了重注入缓冲(reinject buffer)和重传单元(retry unit)。重注入缓冲存储等待重新注入网络的报文,包括重传的报文以及重传报文、确认等控制报文;重传单元为控制单元,负责报文的重新注入以及控制报文的产生,根据内部存储的信息可以生成对应的确认或者重传控制报文。为了实现报文的重传,需要在源节点的重注入缓冲中保留报文的备份。源节点收到重传报文后,会将对应的跨芯粒报文重新注入网络,进行报文的重新传输;源节点收到确认报文后,会清除重注入缓冲中对应报文的备份。

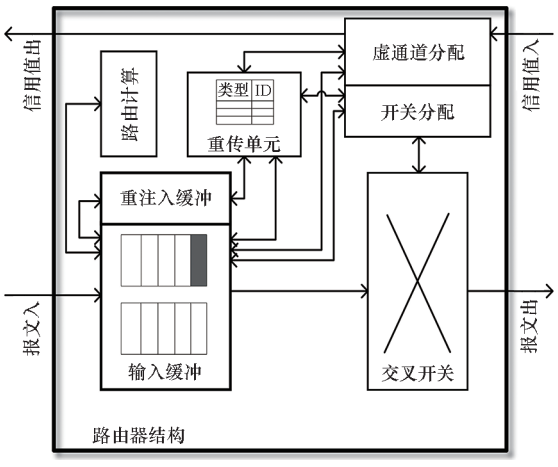


图 4　路由节点架构实现

Fig. 4　Possible implementation of routing node architecture

2　报文转发与报文合并

2.1　报文转发

将暂时无法吸收的跨芯粒报文转发给相邻的边界节点可以提高网络的容错性以及均衡负载。如图 3 所示,P2 到达节点 14 而无法被吸收,则转发给相邻的边界节点 2,假设节点 2 可以吸收,则由节点 2 向源节点 15 返回确认报文。

在报文转发过程中相邻边界节点的选取受到路由算法的转向限制的影响。如图 3 中的 P3,P3 选择的边界节点为 13,不能被吸收时,转发给相邻的节点 14,由于 *XY* 路由策略的限制,P3 并不能直接变换路由路径,否则可能导致死锁。P3 会进入节点 13 的重注入缓冲,重新选择路由路径。报文进入重注入缓冲之后,可以把报文视为新注入网络的报文,重新进行路由计算,因此重新注入的这个过程会改变原有的通道依赖关系,从而避免死锁,报文转发功能的实现基于报文重传。

边界节点分配策略:当一个边界节点出现故障或者拥塞时,采用转发到邻居策略,将报文转发给相邻的边界节点,如图 5 所示。以图 5 中区域 1 的节点为例,区域 1 中跨芯粒的报文需要通过边界节点 1 离开当前芯粒。当报文到达边界节点 1 时,如果不能被吸收,则边界节点 1 负责将报文转发至预先设定的边界节点 2。

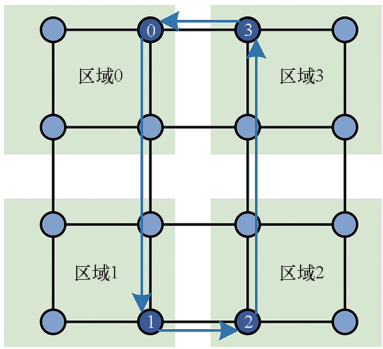


图 5　将报文转发给相邻边界节点的策略

Fig. 5　Strategy for forwarding packets to neighboring boundary node

FTN 策略是介于静态绑定和动态选择之间的边界节点分配策略,优势在于兼顾容错性和负载均衡,并且实现简单,如算法 1 所示。

具体分析如下:在静态绑定策略中,静态绑定的节点对应关系在芯片设计之初确定,可以在综合考虑成本、带宽以及延迟等因素的基础上选择较优方案。该策略易于实现,只需要修改对应的路由信息表。不足之处在于当物理链路出现故障导致边界节点不可用时,需要重新配置静态绑定关系,因此容错性较差,并且可能造成网络负载不均衡。在动态选择策略中,边界节点可以动态选择,即每个路由节点没有固定的边界节点,跨芯粒的报文会根据网络当前状态选取合适的边界节

算法 1　添加报文转发功能的报文重传算法

Alg. 1　Packet retransmission algorithm with packet forward function

输入:报文阻塞时间 $Time_{block}$

输出:报文传输行为

if $Time_{block} \geqslant Block_{threshold}$ **then**//超过阈值
  if $Number_{forward} \geqslant Forward_{threshold}$ **then**
    Return retry to the source node
  else
    Forward to pre-set adjacent nodes
  end if
else
  Wait for switch allocation //等待下一轮仲裁
end if

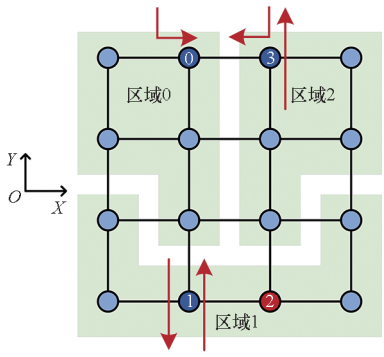
点。优势在于可以避开出现故障或者拥塞的边界节点,负载更加均衡,容错性更好。不足之处在于动态选取的实现难度较大,主要在于报文动态选择边界节点时难以及时获取所有可用的边界节点的状态。

FTN 策略具有一定的灵活性,在链路发生故障时,根据边界节点的状态选择能够正常工作的临近的边界节点。

容错性:报文转发功能可以提供更低成本的容错性,相较于 MTR,不需要重新配置。在 MTR 中,每个路由节点唯一对应一个边界节点。当边界节点出现故障时,会切断部分路由节点跨芯粒报文传输的唯一路径,因而需要重新计算路由信息、分配边界节点,这一过程相对复杂耗时。而在 FTN 的设计中,这一过程可以自发进行,不需要额外的配置过程。报文转发可以将报文转发给相邻的边界节点,从而规避故障链路。

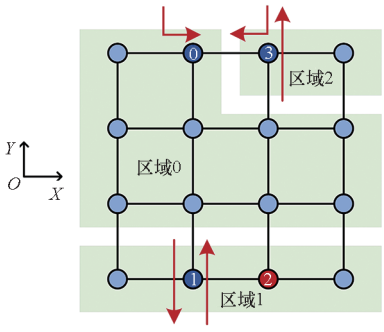
负载均衡性:链路出现故障后,导致可用的边界节点分布不均匀,会使得跨芯粒报文的流量分布不均衡。图 6 展示了出现链路故障后一种可能的负载分布情况。图 6(a)和图 6(b)分别是跨芯粒报文进出的边界节点分布情况,红色箭头表示禁止这个方向的报文,区域 0、1 和 2 表示边界节点负责的范围。

以边界节点 0 的红色箭头为例,跨芯粒的报文进入节点 0 后禁止转向 X 轴正方向。以 6(a)区域 0 为例,所有到达区域 0 内节点的跨芯粒报文都需要经过边界节点 0,因此区域的面积越大,边界节点的负载就越重 。相较于图 2,跨芯粒报文的分布更加集中,使得部分边界节点的负载更重,局部的热点可能会影响网络整体的性能。报文转发首先负载分布更加均匀,其次可以避免重新配置的过程,可以提供更低成本的容错性。



(a) 进入芯粒报文的负载分布情况

(a) Workload distribution for packets entering chiplet



(b) 离开芯粒报文的负载分布情况

(b) Workload distribution for packets leaving chiplet

图 6　一种链路故障下的负载分布情况

Fig. 6　Workload distribution when a link fault occurs

2.2　报文合并

确认报文可以清除报文在重注入缓冲中的备份,使得缓存空间可用。理论上,如果确认报文的平均返回速率(acks/node/cycle)大于报文重注入缓冲的平均写入速率(packets/node/cycle),那么重注入缓冲容量不会对系统性能产生明显影响。因此及时地返回确认报文可以提高重注入缓冲的使用率。在网络负载较低时,重注入缓冲的容量大小没有对系统性能产生明显的影响,由于报文注入率较低,对重注入缓冲的容量大小不敏感。

将若干确认报文请求合并为一个报文,可以减少控制报文的数目,降低网络的负载以及功耗。由于网络负载较低,这种报文合并不会对系统性能造成显著影响。重传单元中存储着生成确认或者重传报文所需的全部信息。正常模式下,控制报文会及时返回,以提高源节点重注入缓冲的利用率。在网络负载较低时,会将同一个目的节点的确认报文合并为一个报文,该报文携带多个 ID 号,可以交互多个报文的确认信息。

3　实验及结果分析

使用 gem5[28] 和 garnet2.0[29] 来搭建片上网络,参数配置如表 1 所示。网络架构如图 1 所示,芯粒内部使用 4×4 的 mesh 网络,使用 *XY* 路由策略[31]。芯粒间再通过 4×4 的 mesh 网络连接,同样使用 *XY* 路由策略。在 MTR 中施加的转向限制如图 2 所示,出现链路故障后转向限制变化如图 6 所示。链路故障在实验中设置为芯粒和中介层连接链路的故障,不考虑芯粒内部和中介层内部的链路故障。在 MTR 中,对报文施加转向限制,可能会导致不均衡的负载,进而影响带宽和延迟,在出现链路故障后,网络可能更加拥挤。重传机制可以提供更加均衡的负载,在引入转发机制后,提升效果更加明显。实验内容主要分为四部分:首先是无故障网络下延迟和带宽的对比;然后是存在链路故障的情况下,延迟以及带宽的变化;接着是网络平均负载的变化;最后是重注入缓冲的容量大小对系统性能的影响。

表 1　系统参数配置

Tab. 1　Configuration of system

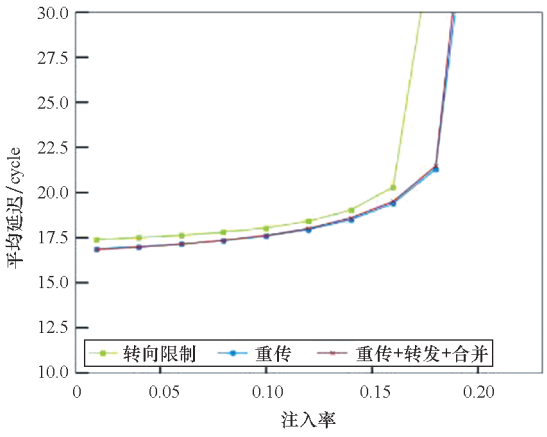
模块	配置
处理器核	x86 ISA, 1.0 GHz
L1 Cache	16 KB Instruction, 32 KB Data
Last Level Cache (LLC)	512 KB
虚网络	3
链路延迟	1 cycle
路由延迟	1 cycle
拓扑	mesh (4×4)
路由算法	*XY*

3.1　规则拓扑

在三种不同流量模式下,延迟及饱和带宽的对比如图 7 所示,相较于 MTR,重传机制具有相似的延迟以及 12.5% ~25% 的饱和带宽提升。报文转发功能主要针对链路故障可能导致的网络不连通或者负载不均衡问题,在不存在链路故障时,网络整体负载较为均匀,无法体现报文转发的功能特点。添加的报文合并功能在网络负载较低时触发,因而图 7 中添加报文合并功能前后的延迟没有出现明显的提升。

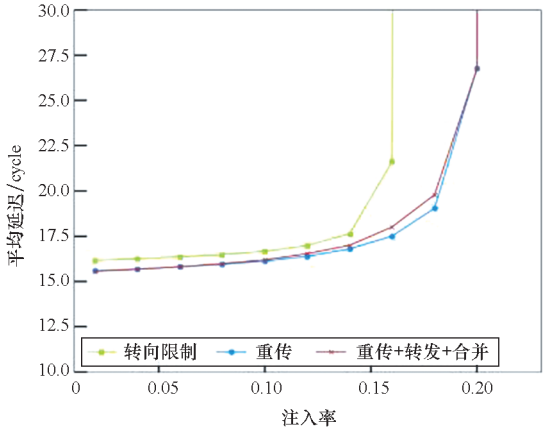
3.2　不规则拓扑

在链路出现故障后的延迟对比如图 8 所示,其中的重传机制包含报文转发和报文合并功能。相较于 MTR,重传机制提供了显著的饱和带宽提升,最高可达 50%。如图 6 所示,不失一般性,假设节点 2 出现故障,无法传输报文。因为可用的边界节点数目的减少,MTR 需要对报文转向进行重新配置。



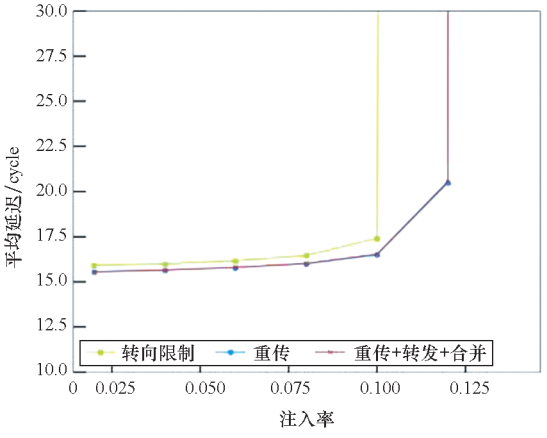
(a) 均匀随机

(a) Uniform random



(b) 混洗

(b) Shuffle



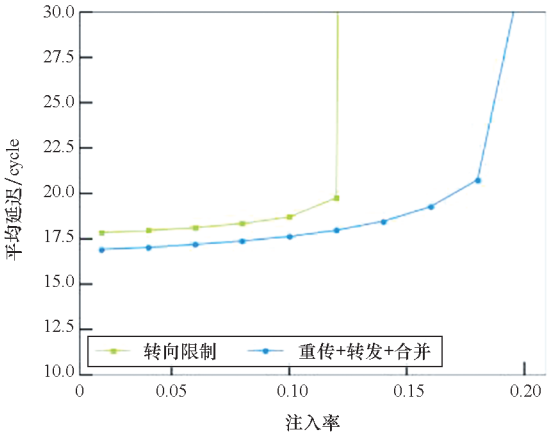
(c) 转置

(c) Transpose

图 7　三种不同流量模式下平均延迟对比

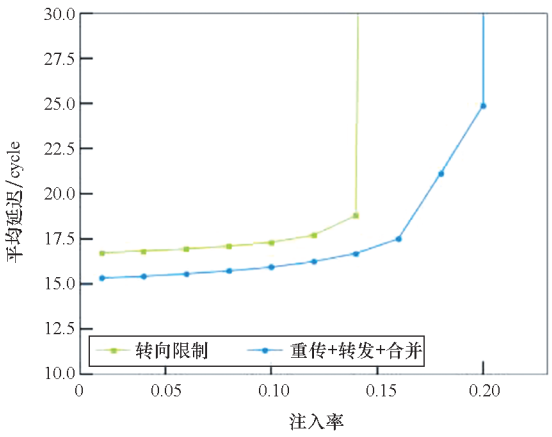
Fig. 7　Comparison of average latency under three traffic patterns

由于边界节点的物理位置无法改变,新的转向限制受到很大的局限,会导致负载的不均衡。图6(b)中,节点0要承担62.5%的流出报文,远高于节点1和节点3的25%和12.5%。不均衡的流量分布会导致局部热点的出现,从而影响饱和带宽。



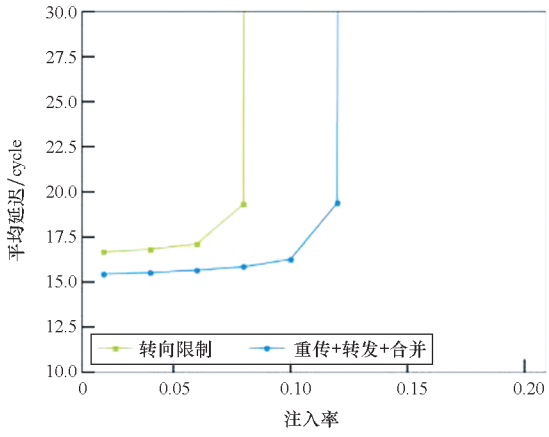
(a) 均匀随机

(a) Uniform random



(b) 混洗

(b) Shuffle



(c) 转置

(c) Transpose

图8 链路故障下三种不同流量模式下平均延迟对比

Fig. 8 Comparison of average latency in three traffic patterns under faulty link

在MTR中,当边界节点出现故障时,会切断部分路由节点跨芯粒报文传输的唯一路径,因而需要重新计算路由信息、分配边界节点,这一过程相对复杂耗时。在重传机制中,不需要额外的配置,这一过程可以自发进行,不需要额外的配置过程。报文转发可以将报文转发给相邻的边界节点,从而规避故障链路。因此,系统本身具有较高的容错性。在图6中,报文转发功能可以将原本属于节点2的报文转发给节点0、1或者3。这样的优势在于可以在避免重新配置的前提下,尽可能均衡整个网络的负载。

3.3 网络负载对确认报文数目的影响

图9展示了确认报文数目的变化,引入报文合并功能后,在网络负载较低时,确认报文的数目有明显的减少,随着网络负载的增加,减少的比例越来越小。确认报文数目的减少,可以降低控制报文的总数,从而降低网络整体的负载。

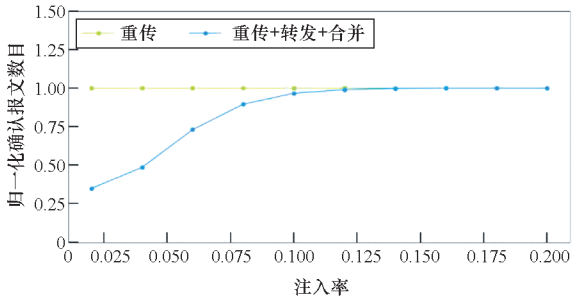


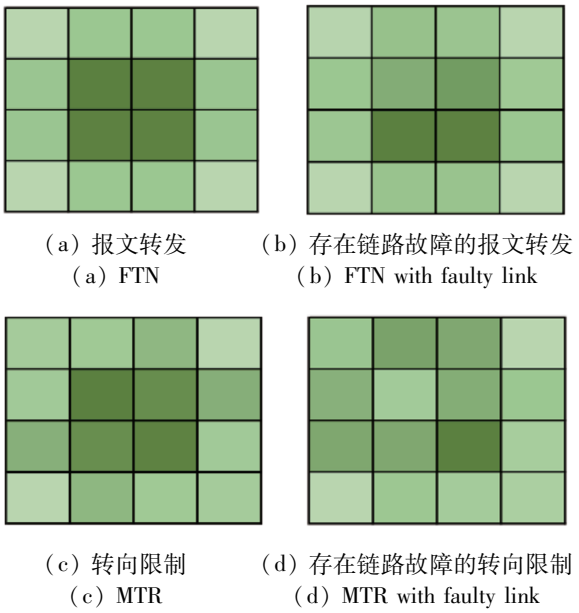
图9 确认报文数目对比

Fig. 9 Comparison of the number of ack packets

图10为网络负载分布热图,展示了跨芯粒报文数目的分布情况,颜色越深代表数目越大。理想状态下,跨芯粒的报文会均匀分布在mesh网络当中。由于mesh网络的对称特性,处于中央位置的节点,会承受更大的负载,但是对称位置节点的负载大致相同,即颜色深浅接近,如图10(a)所示。图10(a)和图10(b)为重传机制下采用FTN策略后的报文分布,分别是没有链路故障和存在链路故障两种场景的结果。

即使在出现链路故障的情况下,报文转发功能也能使报文的分布相对均匀。图10(c)和图10(d)为MTR下的报文分布,分别是没有链路故障和存在链路故障两种场景的结果。由于额外的转向限制,报文的分布相对不均匀,容易形成局部的热点,特别是出现链路故障时。图10(a)和图10(c)对比说明,在正常链路状态下,相较于MTR,FTN的负载更加均衡。图10(b)和图10(d)对比说明,在出现链路故障后,MTR和

FTN 的均衡程度均出现了下降,但是相较于 MTR,FTN 的负载仍然更加均衡。



(a) 报文转发　(b) 存在链路故障的报文转发
(a) FTN　(b) FTN with faulty link

(c) 转向限制　(d) 存在链路故障的转向限制
(c) MTR　(d) MTR with faulty link

图 10　网络负载分布热图
Fig. 10　Heat map of network load

3.4　重注入缓冲的容量

如果源节点的重注入缓冲为满,则跨芯粒的报文不能注入芯粒网络,因而重注入缓冲的容量会对系统性能造成影响。图 11 分析了重注入缓冲的容量对报文延迟的影响。较大的重注入缓冲容量可以降低延迟,但是当容量达到临界值后,延迟降低的效果快速减弱。而且当网络达到过饱和状态后,大的缓存容量反而导致延迟增加。可能是由于较大的重注入缓冲容量会导致重传的报文数目也相应增加,从而进一步加重网络的负担,导致延迟增加。通过物理综合后对比新增加逻辑的晶体管资源开销。假设以重注入缓冲容量为 4 为例,在虚网络为 3,虚通道为 4 的情况下,重注入缓冲仅增加了 5.6% 的缓存资源开销。

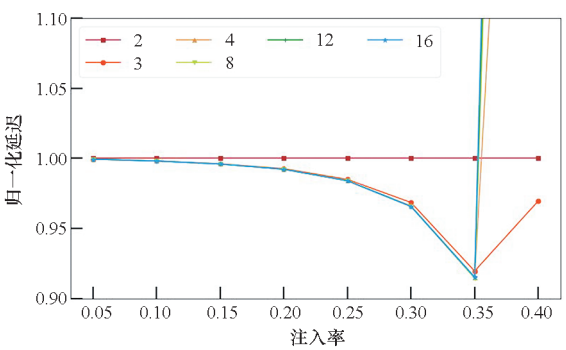


图 11　延迟随着缓存容量的变化
Fig. 11　Normalized latency under different buffer sizes

4　结论

针对之前研究存在的网络延迟和网络带宽受到影响、芯粒内网络边界节点负载不均衡、芯粒间链路发生故障后容错性较差等问题,提出报文重传机制解决跨芯粒的死锁问题,增加报文转发和报文合并功能。实验结果表明,所提方法在网络延迟基本保持不变的前提下提升了网络饱和带宽。相较于转向限制,改进的重传机制具有相似的延迟以及 12.5% ~25% 的饱和带宽提升;在出现链路故障时,饱和带宽的提升最高可达 50%。“报文合并”可以减少控制报文的数目,特别是随着网络负载的增加更加显著,从而降低网络整体的负载。“报文转发”可以自适应处理链路故障,不需要重新配置,容错性更强、成本更低。所提方法在链路故障情况下也能够实现芯粒内网络更均衡的负载。

参考文献(References)

[1] KANNAN A, JERGER N E, LOH G H. Enabling interposer-based disintegration of multi-core processors [C]// Proceedings of the 48th International Symposium on Microarchitecture, 2015.

[2] JERGER N E, KANNAN A, LI Z M, et al. NoC architectures for silicon interposer systems: why pay for more wires when you can get them (from your interposer) for free? [C]// Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture, 2014.

[3] FENG Y X, XIANG D, MA K S. A scalable methodology for designing efficient interconnection network of chiplets[C]// Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2023.

[4] STOW D, AKGUN I, BARNES R, et al. Cost analysis and cost-driven IP reuse methodology for SoC design based on 2.5D/3D integration [C]//Proceedings of the 35th International Conference on Computer-Aided Design, 2016.

[5] HU X, STOW D, XIE Y. Die stacking is happening[J]. IEEE Micro, 2018, 38(1): 22 - 28.

[6] PAL S, PETRISKO D, KUMAR R, et al. Design space exploration for chiplet-assembly-based processors[J]. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 2020, 28(4): 1062 - 1073.

[7] VIVET P, GUTHMULLER E, THONNART Y, et al. 2.3 A 220GOPS 96-core processor with 6 chiplets 3D-stacked on an active interposer offering 0.6 ns/mm latency, 3Tb/s/mm^2 inter-chiplet interconnects and 156 mW/mm^2 @ 82%-peak-efficiency DC-DC converters[C]//Proceedings of the IEEE International Solid-State Circuits Conference(ISSCC), 2020.

[8] XIA J, CHENG C N, ZHOU X P, et al. Kunpeng 920: the first 7-nm chiplet-based 64-core ARM SoC for cloud services[J]. IEEE Micro, 2021, 41(5): 67 - 75.

[9] STOW D, XIE Y, SIDDIQUA T, et al. Cost-effective design of scalable high-performance systems using active and passive interposers[C]//Proceedings of the IEEE/ACM International

Conference on Computer-Aided Design (ICCAD), 2017.

[10] PARK H, KIM J, CHEKURI V C K, et al. Design flow for active interposer-based 2.5-D ICs and study of RISC-V architecture with secure NoC[J]. IEEE Transactions on Components, Packaging and Manufacturing Technology, 2020, 10(12): 2047 – 2060.

[11] WANG T Q, FENG F, XIANG S L, et al. Application defined on-chip networks for heterogeneous chiplets: an implementation perspective[C]//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022.

[12] NAFFZIGER S, BECK N, BURD T, et al. Pioneering chiplet technology and design for the AMD EPYC™ and Ryzen™ processor families: industrial product[C]// Proceedings of the ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA), 2021.

[13] EVERS M, BARNES L, CLARK M. The AMD next-generation "Zen 3" core[J]. IEEE Micro, 2022, 42(3): 7 – 12.

[14] YIN J M, LIN Z F, KAYIRAN O, et al. Modular routing design for chiplet-based systems[C]//Proceedings of the ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), 2018.

[15] MAJUMDER P, KIM S, HUANG J Y, et al. Remote control: a simple deadlock avoidance scheme for modular systems-on-chip[J]. IEEE Transactions on Computers, 2021, 70(11): 1928 – 1941.

[16] WU Y B, WANG L, WANG X H, et al. Upward packet popup for deadlock freedom in modular chiplet-based systems[C]//Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2022.

[17] TAHERI E, PASRICHA S, NIKDAST M. DeFT: a deadlock-free and fault-tolerant routing algorithm for 2.5D chiplet networks[C]//Proceedings of the Design, Automation & Test in Europe Conference & Exhibition (DATE), 2022.

[18] GLASS C J, NI L M. The turn model for adaptive routing[C]//Proceedings of the 19th Annual International Symposium on Computer Architecture(ISCA), 1992.

[19] FARROKHBAKHT H, KAO H, HASAN K, et al. Pitstop: enabling a virtual network free network-on-chip[C]// Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA), 2021.

[20] DAS SHARMA D, PASDAST G, QIAN Z G, et al. Universal chiplet interconnect express (UCIe): an open industry standard for innovations with chiplets at package level[J]. IEEE Transactions on Components, Packaging and Manufacturing Technology, 2022, 12(9): 1423 – 1431.

[21] WU Y B, WANG L, WANG X H, et al. A deflection-based deadlock recovery framework to achieve high throughput for faulty NoCs[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2021, 40(10): 2170 – 2183.

[22] 王永庆, 张民选. K-ary N-cube 中的移动气泡流控策略[J]. 国防科技大学学报, 2012, 34(6): 34 – 38, 53.
WANG Y Q, ZHANG M X. Moveable bubble flow control in K-ary N-cube[J]. Journal of National University of Defense Technology, 2012, 34(6): 34 – 38, 53. (in Chinese)

[23] 肖灿文, 张民选, 过锋. K-ary N-cube 网络中的维度气泡流控与无死锁完全自适应路由[J]. 计算机学报, 2006, 29(5): 801 – 807.
XIAO C W, ZHANG M X, GUO F. DBFC: dimensional bubble flow control with deadlock-free and fully adaptive routing in the K-ary N-cube network[J]. Chinese Journal of Computers, 2006, 29(5): 801 – 807. (in Chinese)

[24] TAHERI E, ISAKOV M, PATOOGHY A, et al. Addressing a new class of reliability threats in 3-D network-on-chips[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2020, 39(7): 1358 – 1371.

[25] TAHERI E, KIM R G, NIKDAST M. AdEle: an adaptive congestion-and-energy-aware elevator selection for partially connected 3D NoCs[C]//Proceedings of the 58th ACM/IEEE Design Automation Conference (DAC), 2021.

[26] LIU Y, GUO R J, XU C Q, et al. A *Q*-learning-based fault-tolerant and congestion-aware adaptive routing algorithm for networks-on-chip[J]. IEEE Embedded Systems Letters, 2022, 14(4): 203 – 206.

[27] EBRAHIMI M, DANESHTALAB M. EbDa: a new theory on design and verification of deadlock-free interconnection networks[C]//Proceedings of the 44th Annual International Symposium on Computer Architecture, 2017.

[28] BINKERT N, BECKMANN B, BLACK G, et al. The gem5 simulator[J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1 – 7.

[29] AGARWAL N, KRISHNA T, PEH L S, et al. GARNET: a detailed on-chip network model inside a full-system simulator[C]// Proceedings of the IEEE International Symposium on Performance Analysis of Systems and Software, 2009.

[30] AUSAVARUNGNIRUN R, FALLIN C, YU X Y, et al. Design and evaluation of hierarchical rings with deflection routing[C]//Proceedings of the IEEE 26th International Symposium on Computer Architecture and High Performance Computing, 2014.

[31] BHARADWAJ S, YIN J, BECKMANN B, et al. Kite: a family of heterogeneous interposer topologies enabled via accurate interconnect modeling[C]//Proceedings of the 57th ACM/IEEE Design Automation Conference (DAC), 2020.